



Europäisches Patentamt
European Patent Office
Office européen des brevets



⑪ Publication number : 0 613 274 A2

⑫

EUROPEAN PATENT APPLICATION

⑬ Application number : 94300677.5

⑮ Int. Cl.⁶ : H04L 29/06

⑭ Date of filing : 28.01.94

⑯ Priority : 29.01.93 US 15366

⑯ Inventor : Sharma, Mohan Byrappa
12148 Jollyville Road No.316

Austin, Texas 78759 (US)

Inventor : Yeung, Yue

11714 D-K Ranch Road

Austin, Texas 78759 (US)

Inventor : Cheng, Chungsiang

10629 Floral Drive

Austin, Texas 78759 (US)

⑯ Date of publication of application :
31.08.94 Bulletin 94/35

⑯ Designated Contracting States :
DE FR GB

⑯ Representative : Moss, Robert Douglas
IBM United Kingdom Limited
Intellectual Property Department
Hursley Park
Winchester Hampshire SO21 2JN (GB)

⑯ Applicant : International Business Machines
Corporation
Old Orchard Road
Armonk, N.Y. 10504 (US)

⑯ Socket structure for concurrent multiple protocol access.

⑯ In a multiprotocol environment, a new socket structure is provided which moves the decision on which protocol to use until the time that the connection is actually made between nodes in the network. The new socket structure is created for every endpoint. All the protocols which could potentially be used to send or receive data is sent a request to create a protocol control block at the time the new socket is created. A selection process determines which of the protocols could be used by the endpoint. The new socket then contains information on each selected protocol. At the time a connection is established, any of the selected protocols could be used. The choice of which protocol to use can be based on user preferences, which protocols are available, the name of the service unit.

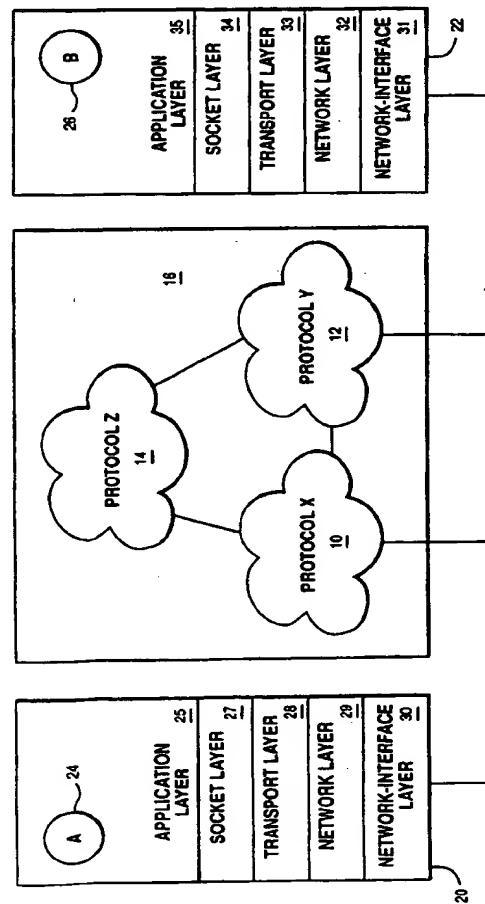


FIG. 1

BACKGROUND OF THE INVENTION

This invention relates generally to data communication in a network of computer systems. More particularly, it relates to a communication endpoint structure which enables application programs resident on systems to communicate through such a network irrespective of the protocol for which the application was written and the protocols available on the network.

Once upon a time, almost all computer systems were standalone processors to which private peripheral devices such as displays, printers and disk drives were connected, acting independently from any other computer system. Yet, it became apparent that there were substantial gains to be made if several computer systems could be cooperatively coupled together. Today, it has become commonplace to couple a multitude of computer systems into a distributed environment such as a local area or wide area network.

However, there are many vendors who have developed their own hardware and software solutions for integrating multiple computer systems. In particular, they have developed different ideas of the format and protocols that should be followed in transporting data through the networks. Some protocols support expedited data transmission by bypassing the standard data flow controls; others require all data to go through the controls. Specialized protocols are used for transport tasks such as establishing and terminating connections between computer systems. Examples of well known communication protocols include System Network Architecture (SNA), Digital Network Architecture (DECnet), Transmission Control Protocol/Internet Protocol (TCP/IP), Network Basic Input/Output System (NetBIOS), Open Systems Interconnection (OSI) and AppleTalk.

Other protocols are known and widely used.

Most distributed application programs are written to an application programming interface (API) which assumes a particular communications protocol. However, the distributed environments which most organizations have assembled are quite complex, comprised of confederations of individual networks running different communication protocols. If the transport protocols assumed by the distributed application's API and the transport protocols actually implemented in one or more of the networks on which the organization would like to send its data are not the same, the use of the application is limited. The problems of such heterogeneity in communications protocols is expected to get worse as more organizations begin to communicate with each other through their networks to perform order processing, direct billing or other cross organization activities.

While the distributed applications could be rewritten so that they can run over each transport protocol or application gateways can be written for each distributed set of distributed applications, the cost of having programmers write all the necessary code makes these approaches economically unattractive. A preferred solution is presented in copending application, Serial No. 731,564, entitled "Compensation for Mismatched Transport Protocols in a Data Communications Network", by R.F. Bird et al, filed July 17, 1991 and hereby incorporated by reference. The incorporated application teaches a transport connection layer between a first transport user at one node in the network and a second transport user at a different node in the network. When the transport protocol assumed by the application at the first node is not available in the network, the data being transferred between the two nodes is automatically altered to be compatible with the available transport protocols. Thus, an organization is able to choose application programs solely on the basis of the functions they provide, rather than the protocols which they require.

The above referenced application teaches a generalized transport layer. One of the transport structures which is presently used in the Berkeley version of the UNIX(TM)environment is called a socket. A socket is an object that identifies a communication endpoint in a network. A socket hides the protocol of the network architecture beneath from the application. A socket allows the association of an endpoint, such as an application program, with one of the selected protocols. This association occurs when the endpoint is created. An endpoint creation implies a static association of the socket with the protocol, which will remain invariant. However, in a multiprotocol environment, as described in the above referenced application, which facilitates heterogeneous connectivity, a transport endpoint may need to be bound to any of several connectivity, a transport endpoint may need to be bound to any of several available protocols after the creation of the endpoint. Therefore, if sockets are to be used in such an environment, a new socket structure which allows dynamic association of the socket with the protocol must be devised. The present invention teaches such a socket structure.

Summary of the Invention

The present invention provides a system and a method for communicating between nodes in a computer network in which a plurality of protocols are useable by network nodes, the method comprising the steps of creating communication endpoint objects defining communication parameters for respective network nodes, the objects having information about the plurality of protocols which are useable by said node for inter-node communication; and at the time communication is requested between said nodes, establishing a connection

between said nodes using a selected one of the plurality of protocols.

The present invention thereby facilitates late binding of an endpoint to a transport protocol in a distributed environment, enabling a more dynamic association between protocol and endpoint.

Preferably, the invention allows native access to a protocol by an application. It is also preferred that the invention provides the necessary information for non-native connections to a protocol.

The invention provides, in a preferred embodiment, a new socket structure which moves the decision on which protocol to use to the time that the connection is actually made between nodes in the network. In a multiprotocol network, the new socket structure can be created for every endpoint. For all of the protocols which could potentially be used to send or receive data a request is made to create a protocol control block at the time the new socket is created. A selection process determines which of the protocols could potentially be used by a given endpoint. The new socket for the endpoint then contains information on each of the selected protocols. At the time a connection is established, any of the selected protocols could be used. Native or non-native connections can be made. The choice of which protocol to use or the order of protocol preference can be based on user preferences through configuration, the local cache, or information from the named service unit.

Brief Description of the Drawings

The invention will now be described in more detail, by way of example, with reference to the accompanying drawings in which:

FIG. 1 is a diagram of two single protocol networks coupled together through a multiprotocol transport network.

FIG. 2 is a block diagram of the transport interface used according to an embodiment of the present invention.

FIG. 3 is a diagram of a conventional socket control block.

FIG. 4 is a diagram of the socket control block according to an embodiment of the present invention.

FIG. 5 is a flow diagram of the creation of a socket according to an embodiment of the present invention.

FIG. 6 is a flow diagram of establishing a connection in a multiprotocol transport network environment, using a multiprotocol socket according to the invention.

FIG. 7 is a representation of a computer system including system unit, keyboard, mouse and display.

FIG. 8 is a block diagram of the computer system components depicted in FIG. 7.

Detailed Description of the Drawings

The following description describes a socket based architecture. However, the invention is not limited to sockets and could be applied to similar communication endpoint objects in other operating systems.

Although the following description contains an adequate disclosure of conventional socket and network architecture to allow one skilled in the art to understand the present invention, the reader is referred to The Design and Implementation of the 4.3 BSD UNIX Operating System by S. J. Leffer et al., 1989 which is hereby incorporated by reference, for a more complete understanding of operating system based on the Berkley version of UNIX™. Such operating systems are well known.

FIG. 1 shows three single protocol networks 10, 12, 14 which are interconnected through a gateway 59 built according to the principles of the present invention. With the growth of network distributed environments, it is not uncommon to see networks using four or five different protocols, such as TCP/IP, NetBIOS, SNA or AppleTALK. Since applications which run on one network will not often run with applications on the other, transport of data throughout the network is hindered. As discussed above, the MultiProtocol Transport Network (MPTN) 16 as taught in the above referenced application addresses these problems by defining an interface and a compensation mechanism to a set of transport services that provide connections across a plurality of transport protocols. By providing a transport boundary between the networks and the applications resident on systems, MPTN provides a common transport interface so that messages from the applications can be transported over any protocol in the network.

As shown in FIG. 1, hosts 20 and 22 are coupled to the multiprotocol transport network 16. While the MPTN 16 appears as though it is a single logical network having a single protocol, host 20 is coupled to a first network 10 which communicates via protocol X, e.g., TCP/IP, and host 22 is coupled to a second network 12 which communicates via protocol Y, e.g., NetBIOS.

Application program A 24 resident in one of the hosts 20 coupled to the MPTN network 16 wishes to communicate to application B 26 resident in another host 22 also coupled to the network 16. Upon application A's call to the socket layer 27, a socket is created by the socket layer 27 defining application A as an endpoint.

Sockets contain information about their type, the supporting protocols used in the transport layer 28 and their state. A connection request goes through the transport layer 28, the network layer 29 and the network interface layer 30 which add the necessary control and data information before the message is sent out on the network. 54. Compensation for the differences between protocol y and X is carried out by the transport layer as taught by the above referenced application.

FIG. 2 depicts the code modules in memory at a computer system which is coupled to the multiprotocol transport network in greater detail. The socket programming interface 60 and common transport semantics 62 correspond to the socket layer and separate the applications 64, 66, 68 from the service drivers 70, 72, 74. Three types of applications 64, 66, 68 are shown in the application layer. To utilize the socket structure of the present invention, NetBIOS applications would be rewritten to the socket API to become the NetBIOS socket application 66. The standard local IPC socket application 64 and TCP/IP applications 68 are already written to a standardized socket programming interface and so would require a minimum of revisions. The common transport semantics 62, include the socket control blocks, which will be described in greater detail below. An interface between the socket layer and the transport layer is comprised by the local IPC service driver 70, the NetBIOS service driver 72 and the INet service driver 74 which correspond to the applications in the application layer. The service drivers are used to emulate the common transport semantics for the transport protocol drivers in the transport layer. In the present invention, they may also contain the compensating means described in the above referenced application which converts a message intended for a first protocol by the application to a second protocol supported by the network. The transport layer includes the NetBIOS 76 and TCP/IP 78 protocol drivers which cause the application message to conform to the protocol format. There is no corresponding local IPC module as it describes a local protocol whose messages are not placed on the network. The network and network interface layers are not pictured, the latter would include device drivers for the I/O hardware which couples the computer system to the network, e.g. a token ring or ethernet driver; the former might include drivers written to the well known NDIS interface.

25 A socket is the object in the Berkeley version of UNIX(TM) from which messages are sent and received. Sockets are created within a communication domain as files are created within a file system. A communication domain summarizes the sets of protocols, the naming conventions, the hardware which may be a particular network and may use an address which refers to the communication domain. The internet domain specified by the address family AF_INET; the NetBIOS domain is referenced by the address family AF_NETBIOS. A connection is a means by which the identity of the sending socket is not required to be sent with each packet of data. The identity of each communication endpoint is exchanged prior to any transmission of data and is maintained at the transmitting and receiving nodes, so that the distributed applications at each end can request the socket information at any time.

35 When an application creates a socket endpoint for a certain protocol and the protocol chosen by the transport network matches that protocol, native networking has occurred. For example, the INet protocol is used to support the INet address family and NetBIOS supports the NetBIOS address family. On the other hand, when the transport protocol does not match the socket endpoint of an application it is termed non-native networking. Using the present invention, however, the application is unaware that a different transport protocol is being used to connect with other nodes in the network.

40 In the present invention, the socket interface is used to connect between replicas of a distributed application or the client and server portions of a client/server application using a variety of transport protocols. The application can select the transport protocol or request that the socket layer determine the protocol. With the non-native networking feature of the present invention, applications written to communicate with one another using one protocol can choose to communicate on another transport protocol which might be optimized for the network environment. For example, an application written for TCP/IP could communicate using the NetBIOS protocol over the network, giving the distributed application a significant performance gain.

50 A conventional socket control block is depicted in FIG. 3, a socket control block 100 contains information about the socket, including send and receive data queues, their type, the supporting protocol, their state and socket identifier. The socket control block 100 includes a protocol switch table pointer 104 and a protocol control block pointer 106. The pointers are used to locate the protocol switch table (not pictured) and the protocol control block 102 respectively. The protocol switch table contains protocol related information, such as entry points to protocol, characteristics of the protocol, certain aspects of its operation which are pertinent to the operation of the socket and so forth. The socket layer interacts with a protocol in the transport layer through the protocol switch table. The user request routine PR_USRREQ is the interface between a protocol and the socket. The protocol control block contains protocol specific information which is used by the protocol and is dependent upon the protocol.

55 A socket control block according to the present invention is shown in FIG. 4. Here, the socket control block 110 is shown broken into two sections, the main socket control block which is largely identical to the conven-

tional socket control block above and the MPTN extension 112 which contains many of the features necessary for the present invention. Both are linked together by a multiprotocol transport network extension 113. Each of the protocols which could be potentially used to send or receive data is sent a request to create a protocol control block 120, 122 at the time the new socket is created. After a selection procedure, the new socket then contains information regarding each selected protocol, the protocol switch table pointer 114, 118, the protocol control block pointer 116, 119, and a pointer to a interface address if applicable for the particular protocol (not pictured). As above, the protocol switch table pointer refers to a respective protocol switch table which defines various entry points for that protocol. Also, the protocol control blocks 120, 122 contain protocol specific information.

At the time of socket creation, no connection is made to any particular protocol to the application endpoint. Connection implies an association of the connection-requestor socket of the requesting application to another connection-acceptor socket. It can be viewed as a communication wire running between the two sockets that are "connected", potentially in two different continents, providing the ability for both of them to send and receive messages. There is no difference between a transmitting socket and a receiving socket for the present invention. After a connection, each can send or receive data.

The decision on which protocol to use is delayed until the time that the connection is actually made. At the time of establishing a connection, any of the network interfaces in a protocol could be used. The order in which the pointers which refer to the protocols and network interfaces is based on user preference, information from the named service unit or the capability of the protocol the socket extension 112 which contains the pointers 114, 116, 118 and 119, also includes state information on the socket and the multiprotocol transport network. The socket extension 112 is used by the socket layer to manage the socket and MPTN states. From the list of available protocols, the socket layer picks those protocols that support the requested socket domain and the type of communication (datagrams, streams, etc), as described in the flow diagram for socket creation described below.

A sample socket control block structure is given in the code below with the following description:
 1. The first 16 bytes of the structure are reserved for alignment and padding.
 2. The next 16 bytes are used for the socket extension 112 which contains the pointers 114, 116, 118 and 119.
 3. The next 16 bytes are used for the multiprotocol transport network extension 113 which contains the pointers 114, 116, 118 and 119.
 4. The next 16 bytes are used for the protocol control block 120 which contains the pointers 116 and 119.
 5. The next 16 bytes are used for the protocol control block 122 which contains the pointers 116 and 119.
 6. The next 16 bytes are used for the interface address pointer 116.
 7. The next 16 bytes are used for the interface address pointer 119.
 8. The next 16 bytes are used for the interface address pointer 118.
 9. The next 16 bytes are used for the interface address pointer 114.
 10. The next 16 bytes are used for the named service unit pointer 112.
 11. The next 16 bytes are used for the named service unit pointer 111.
 12. The next 16 bytes are used for the named service unit pointer 110.
 13. The next 16 bytes are used for the named service unit pointer 109.
 14. The next 16 bytes are used for the named service unit pointer 108.
 15. The next 16 bytes are used for the named service unit pointer 107.
 16. The next 16 bytes are used for the named service unit pointer 106.
 17. The next 16 bytes are used for the named service unit pointer 105.
 18. The next 16 bytes are used for the named service unit pointer 104.
 19. The next 16 bytes are used for the named service unit pointer 103.
 20. The next 16 bytes are used for the named service unit pointer 102.
 21. The next 16 bytes are used for the named service unit pointer 101.
 22. The next 16 bytes are used for the named service unit pointer 100.
 23. The next 16 bytes are used for the named service unit pointer 99.
 24. The next 16 bytes are used for the named service unit pointer 98.
 25. The next 16 bytes are used for the named service unit pointer 97.
 26. The next 16 bytes are used for the named service unit pointer 96.
 27. The next 16 bytes are used for the named service unit pointer 95.
 28. The next 16 bytes are used for the named service unit pointer 94.
 29. The next 16 bytes are used for the named service unit pointer 93.
 30. The next 16 bytes are used for the named service unit pointer 92.
 31. The next 16 bytes are used for the named service unit pointer 91.
 32. The next 16 bytes are used for the named service unit pointer 90.
 33. The next 16 bytes are used for the named service unit pointer 89.
 34. The next 16 bytes are used for the named service unit pointer 88.
 35. The next 16 bytes are used for the named service unit pointer 87.
 36. The next 16 bytes are used for the named service unit pointer 86.
 37. The next 16 bytes are used for the named service unit pointer 85.
 38. The next 16 bytes are used for the named service unit pointer 84.
 39. The next 16 bytes are used for the named service unit pointer 83.
 40. The next 16 bytes are used for the named service unit pointer 82.
 41. The next 16 bytes are used for the named service unit pointer 81.
 42. The next 16 bytes are used for the named service unit pointer 80.
 43. The next 16 bytes are used for the named service unit pointer 79.
 44. The next 16 bytes are used for the named service unit pointer 78.
 45. The next 16 bytes are used for the named service unit pointer 77.
 46. The next 16 bytes are used for the named service unit pointer 76.
 47. The next 16 bytes are used for the named service unit pointer 75.
 48. The next 16 bytes are used for the named service unit pointer 74.
 49. The next 16 bytes are used for the named service unit pointer 73.
 50. The next 16 bytes are used for the named service unit pointer 72.
 51. The next 16 bytes are used for the named service unit pointer 71.
 52. The next 16 bytes are used for the named service unit pointer 70.
 53. The next 16 bytes are used for the named service unit pointer 69.
 54. The next 16 bytes are used for the named service unit pointer 68.
 55. The next 16 bytes are used for the named service unit pointer 67.
 56. The next 16 bytes are used for the named service unit pointer 66.
 57. The next 16 bytes are used for the named service unit pointer 65.
 58. The next 16 bytes are used for the named service unit pointer 64.
 59. The next 16 bytes are used for the named service unit pointer 63.
 60. The next 16 bytes are used for the named service unit pointer 62.
 61. The next 16 bytes are used for the named service unit pointer 61.
 62. The next 16 bytes are used for the named service unit pointer 60.
 63. The next 16 bytes are used for the named service unit pointer 59.
 64. The next 16 bytes are used for the named service unit pointer 58.
 65. The next 16 bytes are used for the named service unit pointer 57.
 66. The next 16 bytes are used for the named service unit pointer 56.
 67. The next 16 bytes are used for the named service unit pointer 55.
 68. The next 16 bytes are used for the named service unit pointer 54.
 69. The next 16 bytes are used for the named service unit pointer 53.
 70. The next 16 bytes are used for the named service unit pointer 52.
 71. The next 16 bytes are used for the named service unit pointer 51.
 72. The next 16 bytes are used for the named service unit pointer 50.
 73. The next 16 bytes are used for the named service unit pointer 49.
 74. The next 16 bytes are used for the named service unit pointer 48.
 75. The next 16 bytes are used for the named service unit pointer 47.
 76. The next 16 bytes are used for the named service unit pointer 46.
 77. The next 16 bytes are used for the named service unit pointer 45.
 78. The next 16 bytes are used for the named service unit pointer 44.
 79. The next 16 bytes are used for the named service unit pointer 43.
 80. The next 16 bytes are used for the named service unit pointer 42.
 81. The next 16 bytes are used for the named service unit pointer 41.
 82. The next 16 bytes are used for the named service unit pointer 40.
 83. The next 16 bytes are used for the named service unit pointer 39.
 84. The next 16 bytes are used for the named service unit pointer 38.
 85. The next 16 bytes are used for the named service unit pointer 37.
 86. The next 16 bytes are used for the named service unit pointer 36.
 87. The next 16 bytes are used for the named service unit pointer 35.
 88. The next 16 bytes are used for the named service unit pointer 34.
 89. The next 16 bytes are used for the named service unit pointer 33.
 90. The next 16 bytes are used for the named service unit pointer 32.
 91. The next 16 bytes are used for the named service unit pointer 31.
 92. The next 16 bytes are used for the named service unit pointer 30.
 93. The next 16 bytes are used for the named service unit pointer 29.
 94. The next 16 bytes are used for the named service unit pointer 28.
 95. The next 16 bytes are used for the named service unit pointer 27.
 96. The next 16 bytes are used for the named service unit pointer 26.
 97. The next 16 bytes are used for the named service unit pointer 25.
 98. The next 16 bytes are used for the named service unit pointer 24.
 99. The next 16 bytes are used for the named service unit pointer 23.
 100. The next 16 bytes are used for the named service unit pointer 22.
 101. The next 16 bytes are used for the named service unit pointer 21.
 102. The next 16 bytes are used for the named service unit pointer 20.
 103. The next 16 bytes are used for the named service unit pointer 19.
 104. The next 16 bytes are used for the named service unit pointer 18.
 105. The next 16 bytes are used for the named service unit pointer 17.
 106. The next 16 bytes are used for the named service unit pointer 16.
 107. The next 16 bytes are used for the named service unit pointer 15.
 108. The next 16 bytes are used for the named service unit pointer 14.
 109. The next 16 bytes are used for the named service unit pointer 13.
 110. The next 16 bytes are used for the named service unit pointer 12.
 111. The next 16 bytes are used for the named service unit pointer 11.
 112. The next 16 bytes are used for the named service unit pointer 10.
 113. The next 16 bytes are used for the named service unit pointer 9.
 114. The next 16 bytes are used for the named service unit pointer 8.
 115. The next 16 bytes are used for the named service unit pointer 7.
 116. The next 16 bytes are used for the named service unit pointer 6.
 117. The next 16 bytes are used for the named service unit pointer 5.
 118. The next 16 bytes are used for the named service unit pointer 4.
 119. The next 16 bytes are used for the named service unit pointer 3.
 120. The next 16 bytes are used for the named service unit pointer 2.
 121. The next 16 bytes are used for the named service unit pointer 1.
 122. The next 16 bytes are used for the named service unit pointer 0.

```

/* socketva.h.....
 * Kernel structure per socket.
 * Contains send and receive buffer queues.
 * handle on protocol and pointer to protocol
 * private data, error information and MPTN extensions.
 * The first part of this structure ( upto the so_mptn field ) is identical
 * to the BSD4.3 socket control block.
 */
5      struct socket {
        short so_type;           /* generic type, see socket.h */
        short so_options;        /* from socket call, see socket.h */
        short so_linger;         /* time to linger while closing */
        short so_state;          /* internal state flags SS *, below */
        caddr_t so_pcb;          /* protocol control block */
        struct protosw far *so_proto; /* protocol handle */
    /*
10     * Variables for connection queueing.
     * Socket where accepts occur is so_head in all subsidiary sockets.
     * If so_head is 0, socket is not related to an accept.
     * For head socket so_g0 queues partially completed connections.
     * while so_q is a queue of connections ready to be accepted.
     * If a connection is aborted and it has so_head set, then
     * it has to be pulled out of either so_g0 or so_q.
     * We allow connections to queue up based on current queue lengths
     * and limit on number of queued connections for this socket.
    */
20     struct socket far *so_head; /* back pointer to accept socket */
        struct socket far *so_g0; /* queue of partial connections */
        struct socket far *so_q; /* queue of incoming connections */
        short so_golen;          /* partials on so_g0 */
        short so_glen;           /* number of connections on so_q */
        short so_glimit;          /* max number queued connections */
        short so_timeo;           /* connection timeout */
        u_short so_error;          /* error affecting connection */
        short so_pgrp;           /* process group (so_pgrp for signals */
        u_long so_cobmark;        /* chars to oob mark */
30     /* Variables for socket buffering */
    */
35     struct sockbuf {
        u_long sb_cc;           /* actual chars in buffer */
        u_long sb_hiwat;         /* max actual char count */
        u_long sb_abcnt;         /* chars of mbufs used */
        u_long sb_sbmax;         /* max chars of mbufs to use */
        u_long sb_lowat;         /* low water mark (not used yet) */
        struct mbuf far *sb_mb; /* the mbuf chain */
        struct proc far *sb_sei; /* process selecting read/write */
        short sb_timeo;          /* timeout (not used yet) */
        short sb_flags;           /* flags, see below */
    }
40     } so_rcv, so_snd;
    /* MPTN SOCKET EXTENSION */

        struct m_esock far * so_mptn; /* socket MPTN extensions */

45     #define SB_MAX    ((u_long)(64*1024L)) /* max chars in sockbuf */
#define SB_LOCK    0x01 /* lock on data queue (so_rcv only) */
#define SB_WANT    0x02 /* someone is waiting to lock */
#define SB_WAIT    0x04 /* someone is waiting for data/space */
#define SB_SEL     0x08 /* buffer is selected */
#define SB_COLL    0x10 /* collision selecting */
    };

50     /* Socket extensions for MPTN
     * The socket control block points to this structure which contains
     * all the MPTN related socket extensions.
     * Since m_esock, the MPTN extensions to sockets, is contained in one
     * mbuf, we could use the rest of mbuf space to accommodate the pcb
     * pointers.
    */
55

```

```

/*
 * defines the structure for storing additional pointers... used in m_sock.
 * The structure m_addr defines the address format. It is defined in mptndef.h.
 * The structure m_info defines the user specified connection characteristics
 * and is defined in mptndef.h.
 * The structure bnlst defines the user configured protocol preference list
 * and is defined in mptndef.h.
 */
5
10
15
20
25
30
35
40
45
50
55
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
7510
7511
7512
7513
7514
7515
7516
7517
7518
7519
7520
7521
7522
7523
7524
7525
7526
7527
7528
7529
7530
7531
7532
7533
7534
7535
7536
7537
7538
7539
7540
7541
7542
7543
7544
7545
7546
7547
7548
7549
7550
7551
7552
7553
7554
7555
7556
7557
7558
7559
75510
75511
75512
75513
75514
75515
75516
75517
75518
75519
75520
75521
75522
75523
75524
75525
75526
75527
75528
75529
75530
75531
75532
75533
75534
75535
75536
75537
75538
75539
75540
75541
75542
75543
75544
75545
75546
75547
75548
75549
75550
75551
75552
75553
75554
75555
75556
75557
75558
75559
75560
75561
75562
75563
75564
75565
75566
75567
75568
75569
75570
75571
75572
75573
75574
75575
75576
75577
75578
75579
75580
75581
75582
75583
75584
75585
75586
75587
75588
75589
755810
755811
755812
755813
755814
755815
755816
755817
755818
755819
755820
755821
755822
755823
755824
755825
755826
755827
755828
755829
755830
755831
755832
755833
755834
755835
755836
755837
755838
755839
755840
755841
755842
755843
755844
755845
755846
755847
755848
755849
755850
755851
755852
755853
755854
755855
755856
755857
755858
755859
755860
755861
755862
755863
755864
755865
755866
755867
755868
755869
755870
755871
755872
755873
755874
755875
755876
755877
755878
755879
755880
755881
755882
755883
755884
755885
755886
755887
755888
755889
755890
755891
755892
755893
755894
755895
755896
755897
755898
755899
7558100
7558110
7558120
7558130
7558140
7558150
7558160
7558170
7558180
7558190
7558200
7558210
7558220
7558230
7558240
7558250
7558260
7558270
7558280
7558290
7558300
7558310
7558320
7558330
7558340
7558350
7558360
7558370
7558380
7558390
7558400
7558410
7558420
7558430
7558440
7558450
7558460
7558470
7558480
7558490
7558500
7558510
7558520
7558530
7558540
7558550
7558560
7558570
7558580
7558590
7558600
7558610
7558620
7558630
7558640
7558650
7558660
7558670
7558680
7558690
7558700
7558710
7558720
7558730
7558740
7558750
7558760
7558770
7558780
7558790
7558800
7558810
7558820
7558830
7558840
7558850
7558860
7558870
7558880
7558890
7558900
7558910
7558920
7558930
7558940
7558950
7558960
7558970
7558980
7558990
75581000
75581100
75581200
75581300
75581400
75581500
75581600
75581700
75581800
75581900
75582000
75582100
75582200
75582300
75582400
75582500
75582600
75582700
75582800
75582900
75583000
75583100
75583200
75583300
75583400
75583500
75583600
75583700
75583800
75583900
75584000
75584100
75584200
75584300
75584400
75584500
75584600
75584700
75584800
75584900
75585000
75585100
75585200
75585300
75585400
75585500
75585600
75585700
75585800
75585900
75586000
75586100
75586200
75586300
75586400
75586500
75586600
75586700
75586800
75586900
75587000
75587100
75587200
75587300
75587400
75587500
75587600
75587700
75587800
75587900
75588000
75588100
75588200
75588300
75588400
75588500
75588600
75588700
75588800
75588900
75589000
75589100
75589200
75589300
75589400
75589500
75589600
75589700
75589800
75589900
755810000
755811000
755812000
755813000
755814000
755815000
755816000
755817000
755818000
755819000
755820000
755821000
755822000
755823000
755824000
755825000
755826000
755827000
755828000
755829000
755830000
755831000
755832000
755833000
755834000
755835000
755836000
755837000
755838000
755839000
755840000
755841000
755842000
755843000
755844000
755845000
755846000
755847000
755848000
755849000
755850000
755851000
755852000
755853000
755854000
755855000
755856000
755857000
755858000
755859000
755860000
755861000
755862000
755863000
755864000
755865000
755866000
755867000
755868000
755869000
755870000
755871000
755872000
755873000
755874000
755875000
755876000
755877000
755878000
755879000
755880000
755881000
755882000
755883000
755884000
755885000
755886000
755887000
755888000
755889000
755890000
755891000
755892000
755893000
755894000
755895000
755896000
755897000
755898000
755899000
7558100000
7558110000
7558120000
7558130000
7558140000
7558150000
7558160000
7558170000
7558180000
7558190000
7558200000
7558210000
7558220000
7558230000
7558240000
7558250000
7558260000
7558270000
7558280000
7558290000
7558300000
7558310000
7558320000
7558330000
7558340000
7558350000
7558360000
7558370000
7558380000
7558390000
7558400000
7558410000
7558420000
7558430000
7558440000
7558450000
7558460000
7558470000
7558480000
7558490000
7558500000
7558510000
7558520000
7558530000
7558540000
7558550000
7558560000
7558570000
7558580000
7558590000
7558600000
7558610000
7558620000
7558630000
7558640000
7558650000
7558660000
7558670000
7558680000
7558690000
7558700000
7558710000
7558720000
7558730000
7558740000
7558750000
7558760000
7558770000
7558780000
7558790000
7558800000
7558810000
7558820000
7558830000
7558840000
7558850000
7558860000
7558870000
7558880000
7558890000
7558900000
7558910000
7558920000
7558930000
7558940000
7558950000
7558960000
7558970000
7558980000
7558990000
75581000000
75581100000
75581200000
75581300000
75581400000
75581500000
75581600000
75581700000
75581800000
75581900000
75582000000
75582100000
75582200000
75582300000
75582400000
75582500000
75582600000
75582700000
75582800000
75582900000
75583000000
75583100000
75583200000
75583300000
75583400000
75583500000
75583600000
75583700000
75583800000
75583900000
75584000000
75584100000
75584200000
75584300000
75584400000
75584500000
75584600000
75584700000
75584800000
75584900000
75585000000
75585100000
75585200000
75585300000
75585400000
75585500000
75585600000
75585700000
75585800000
75585900000
75586000000
75586100000
75586200000
75586300000
75586400000
75586500000
75586600000
75586700000
75586800000
75586900000
75587000000
75587100000
75587200000
75587300000
75587400000
75587500000
75587600000
75587700000
75587800000
75587900000
75588000000
75588100000
75588200000
75588300000
75588400000
75588500000
75588600000
75588700000
75588800000
75588900000
75589000000
75589100000
75589200000
75589300000
75589400000
75589500000
75589600000
75589700000
75589800000
75589900000
755810000000
755811000000
755812000000
755813000000
755814000000
755815000000
755816000000
755817000000
755818000000
755819000000
755820000000
755821000000
755822000000
755823000000
755824000000
755825000000
755826000000
755827000000
755828000000
755829000000
755830000000
755831000000
755832000000
755833000000
755834000000
755835000000
755836000000
755837000000
755838000000
755839000000
755840000000
755841000000
755842000000
755843000000
755844000000
755845000000
755846000000
755847000000
755848000000
755849000000
755850000000
755851000000
755852000000
755853000000
755854000000
755855000000
755856000000
755857000000
755858000000
755859000000
755860000000
755861000000
755862000000
755863000000
755864000000
755865000000
755866000000
755867000000
755868000000
755869000000
755870000000
755871000000
755872000000
755873000000
755874000000
755875000000
755876000000
755877000000
755878000000
755879000000
755880000000
755881000000
755882000000
755883000000
755884000000
755885000000
755886000000
755887000000
755888000000
755889000000
755890000000
755891000000
755892000000
755893000000
755894000000
755895000000
755896000000
755897000000
755898000000
755899000000
7558100000000
7558110000000
7558120000000
7558130000000
7558140000000
7558150000000
7558160000000
7558170000000
7558180000000
7558190000000
7558200000000
7558210000000
7558220000000
7558230000000
7558240000000
7558250000000
7558260000000
7558270000000
7558280000000
7558290000000
7558300000000
7558310000000
7558320000000
7558330000000
7558340000000
7558350000000
7558360000000
7558370000000
7558380000000
7558390000000
7558400000000
7558410000000
7558420000000
7558430000000
7558440000000
7558450000000
7558460000000
7558470000000
7558480000000
7558490000000
7558500000000
7558510000000
7558520000000
7558530000000
7558540000000
7558550000000
7558560000000
7558570000000
7558580000000
7558590000000
7558600000000
7558610000000
7558620000000
7558630000000
7558640000000
7558650000000
7558660000000
7558670000000
7558680000000
7558690000000
7558700000000
7558710000000
7558720000000
7558730000000
7558740000000
7558750000000
7558760000000
7558770000000
7558780000000
7558790000000
7558800000000
7558810000000
7558820000000
7558830000000
7558840000000
7558850000000
7558860000000
7558870000000
7558880000000
7558890000000
7558900000000
7558910000000
7558920000000
7558930000000
7558940000000
7558950000000
7558960000000
7558970000000
7558980000000
7
```

```

* Socket state bits
*/
5  #define SS_NOFDREF 0x001 /* no file table ref any more */
#define SS_ISCONNECTED 0x002 /* socket connected to a peer */
#define SS_ISCONNECTING 0x004 /* in process of connecting to peer */
#define SS_ISDISCONNECTING 0x008 /* in process of disconnecting */
#define SS_CANTSENDMORE 0x010 /* can't send more data to peer */
#define SS_CANTRCVMORE 0x020 /* can't receive more data from peer */
#define SS_RCVATMARK 0x040 /* at mark on input */

10 #define SS_PRIV 0x080 /* privileged for broadcast, raw... */
#define SS_NBIO 0x100 /* non-blocking ops */
#define SS_ASYNC 0x200 /* async i/o notify */

#define SS_CANCEL 0x4000 /* cancel call */
#define SS_PUSHSEN 0x8000 /* seen push */

15 /* Rest of the info is the same as in the BSD4.3 socketva.h */
Copyright, IBM Corp. 1993

```

Conventional socket creation starts with a call to the socket API. A domain table is searched for the address family, the type and protocol which is desired by the application. If a match is found the protocol switch table entry is set in the socket control block. Next, the user requests entry and the selected protocol is called to create of the protocol control block. If a match is not found in the domain table for the address family, type and protocol desired by the application, an error is returned to the application.

FIG. 5 depicts the process of creating a socket according to the present invention. The process begins with a socket creation request 150 from an application to the socket API, the application wishes to send or receive data across the network. The command to the socket API for the request takes the form of socket=(AF, *, type, proto). "AF" refers to the address family and communication domain; "type" refers to one of the socket types defined in a system header file. The "proto" or protocol parameter is used to indicate that a specific protocol is to be used. A test is performed in step 152 to determine if "proto" is specified. If so, the normal socket creation process in step 154 is carried out and the process ends, step 155.

If "proto" is not specified, the process continues to create a multiprotocol socket. Each protocol is associated with a protocol switch table. For each protocol switch table, step 155, tests are performed whether the type and address family of the requesting endpoint, steps 158 and 160, are matched by the protocol. In step 158 the test for "type" match is performed. If the test fails, the process returns for the next protocol switch table, step 156. If the test succeeds, the test for address family match is performed in step 160. Both the "type" and "family" are represented as integers. By matching, the socket layer compares the "type" and "address family" field supplied by the user and the "type" and "address family" fields in the protocol. If a match is found for both type and address family, the protocol is selected as a candidate protocol and set in the socket extension with pointers to the protocol switch table and protocol control block, step 162. If there is no match, the process determines whether the address family is supported non-natively in the network, step 164. If the protocol is supported, step 166, the protocol is selected as a candidate protocol and set in the socket extension with pointers to the protocol switch table and protocol control block, step 162. If the protocol is not supported, in step 166, a test is performed to determine whether it is the last protocol switch table. If not, the process returns to step 156. If it is the last protocol switch table, the process ends, step 170. The protocol pointers can be reordered within the extension according to the application or user-preference through the use of a configuration tool or according to information from the named server.

In FIG. 6, the process to establish a connection using the multiprotocol socket is described. The process begins in step 200 where the socket layer tries connecting to a specified destination using the first protocol from the list of protocols in the socket extension. The choice of which of the protocols to try first can be based on the configuration of user specified preference order of protocols. If the connection succeeds, step 222, the process ends in step 223. If not, the next protocol is used to try to establish a connection in step 224. Tests are performed determine whether a connection is made, step 226. If so, the process ends. If a connection is not made, a test is performed to determine whether it is the last protocol in the socket extension. If not, the next protocol in the extension is tested, step 224. If all the protocols used when creating the socket have failed to provide the connection, the transport provider returns a notification of failure to the application, step 232.

As mentioned previously, the invention finds use in a plurality of computers which are part of a network such as a Local Area Network or wide Area Network. Although the specific choice of computer in these networks is limited only by performance and storage requirements, computers in the IBM PS/2 (TM) series of computers could be used in the present invention. For additional information on IBM's PS/2 series of computers,

the reader is referred to Technical Reference Manual Personal Systems/2 Model 50, 60 Systems IBM Corporation, Part No. 68X2224 Order Number 568X-2224 and Technical Reference Manual Personal Systems/2 (Model 80) IBM Corporation Part No. 68X 2256 Order Number S68X-2254. One operating system which an IBM PS/2 personal computer may run is IBM's OS/2 2.0 (TM). For more information on the IBM OS/2 2.0 Operating System, the reader is referred to OS/2 2.0 Technical Library Programming guide Vol. 1, 2, 3 Version 2.00 Order Nos. 10G6261, 10G6495, 10G6494. In the alternative, the computer system might be in the IBM RISC System/6000 (TM) line of computers which run on the AIX (TM) operating system. The various models of the RISC System/6000 are described in many publications of the IBM Corporation, for example, RISC System/6000, 7073 and 7016 POWERstation and POWERserver Hardware Technical reference, Order No. SA23-2644-00. The AIX operating system is described in General Concepts and Procedure--AIX Version 3 for RISC System/6000 Order No. SC23-2202-00 as well as other publications of the IBM Corporation. In lieu of the cited references, the reader is offered the following general description of a computer system which could be utilized to practice the present invention.

In FIG. 7, a computer 310, comprising a system unit 311, a keyboard 312, a mouse 313 and a display 314 are depicted. The screen 316 of display device 314 is used to present the visual changes to the data object. The graphical user interface supported by the operating system allows the user to use a point and shoot method of input by moving the pointer 315 to an icon representing a data object at a particular location on the screen 316 and press one of the mouse buttons to perform a user command or selection.

FIG. 8 shows a block diagram of the components of the computer shown in FIG. 7. The system unit 11 includes a system bus or plurality of system buses 21 to which various components are coupled and by which communication between the various components is accomplished. The microprocessor 322 is connected to the system bus 321 and is supported by read only memory (ROM) 323 and random access memory (RAM) 324 also connected to system bus 321. A microprocessor in the IBM multimedia PS/2 series of computers is one of the Intel family of microprocessors including the 386 or 486 microprocessors. However, other microprocessors included, but not limited to, Motorola's family of microprocessors such as the 68000, 68020 or the 68030 microprocessors and various Reduced Instruction Set Computer (RISC) microprocessors manufactured by IBM, Hewlett Packard, Sun, Intel, Motorola and others may be used in the specific computer.

The ROM 323 contains among other code the Basic Input-Output system (BIOS) which controls basic hardware operations such as the interaction and the disk drives and the keyboard. The RAM 324 is the main memory into which the operating system, transport providers and application programs are loaded. The memory management chip 325 is connected to the system bus 321 and controls direct memory access operations including, passing data between the RAM 324 and hard disk drive 326 and floppy disk drive 327. The CD ROM 332 is also coupled to the system bus 321.

Also connected to the system bus 321 are various I/O controllers: The keyboard controller 328, the mouse controller 329, the video controller 330, and the audio controller 331 which control their respective hardware, the keyboard 312, the mouse 313, the display 314 and the speaker 315. Also coupled to the system bus 321 is the digital signal processor 333 which controls the sound produced by the speaker 315 and is preferably incorporated into the audio controller 331. An I/O controller 340 such as a Token Ring Adapter enables communication over a network 342 to other similarly configured data processing systems.

While the invention has been described with respect to particular embodiments above, it will be understood by those skilled in the art that modifications may be made without departing from the spirit and scope of the present invention. The preceding description has described the principles of the present invention in terms of a particular communications endpoint object, a socket, in the socket layer between the application layer and transport layer. The principles of the invention could be extended to provide similarly configured communication endpoint objects in other layers. For example, an object in the network interface layer could be used to monitor a plurality of underlying MAC drivers so that data could be sent or received over a plurality of MAC protocols to different types of networks. These embodiments are for purposes of example and illustration only and are not to be taken to limit the scope of the invention narrower than the scope of the appended claims.

50 **Claims**

1. A method for communicating between nodes in a computer network in which a plurality of protocols are useable by network nodes, the method comprising the steps of:
55 creating communication endpoint objects defining communication parameters for respective network nodes, the objects having information about the plurality of protocols which are useable by said node for inter-node communication; and
at the time communication is requested between said nodes, establishing a connection between

said nodes using a selected one of the plurality of protocols.

2. A method according to claim 1, wherein the selected protocol is chosen based on user preferences.
- 5 3. A method according to claim 1, wherein the selected protocol is chosen based on a capability of the selected protocol.
4. A method according to any one of the preceding claims, wherein the creating step comprises the steps of:
 - 10 requesting information of each of the plurality of protocols;
 - selecting a set of protocols from the plurality of protocols, which selected protocols are useable by the respective nodes;
 - building a protocol control block for each of the selected protocols; and,
 - inserting a pointer in the communication endpoint object for each protocol control block of a selected protocol.
- 15 5. A method according to claim 4, wherein the order in which the pointers are inserted is based on user preference.
6. A method according to claim 4 wherein the selecting step comprises the steps of:
 - 20 determining whether there is a match for a first protocol for a first set of protocol parameters, the first protocol corresponding to a first pointer in the socket; and,
 - responsive to finding no match for the first protocol, determining whether the first protocol is supported non-natively in the network.
- 25 7. A system for communicating between nodes in a computer network in which a plurality of protocols are useable by network nodes, the system comprising:
 - means for creating communication endpoint objects defining communication parameters for respective network nodes, which objects have information about the plurality of protocols useable by respective computer systems each coupled to one or more nodes in the network; and,
 - 30 means for establishing a connection between a first and a second computer system using a selected one of the plurality of protocols at the time communication is requested between nodes on the different systems.
8. A system according to claim 7, which further comprises:
 - 35 means for selecting a set of protocols from said plurality of protocols;
 - means for requesting information from each of the plurality of protocols;
 - means for building a protocol control block for each of the selected protocols; and,
 - means for inserting a pointer in the communication endpoint object for each protocol control block for a selected protocol.
- 40 9. A system according to claim 8, which further comprises:
 - means for determining whether there is a match for a first protocol for a first set of protocol parameters, the first protocol corresponding to a first pointer in the socket; and,
 - means for determining, responsive to finding no match for the first protocol, whether the first protocol is supported non-natively in the network.

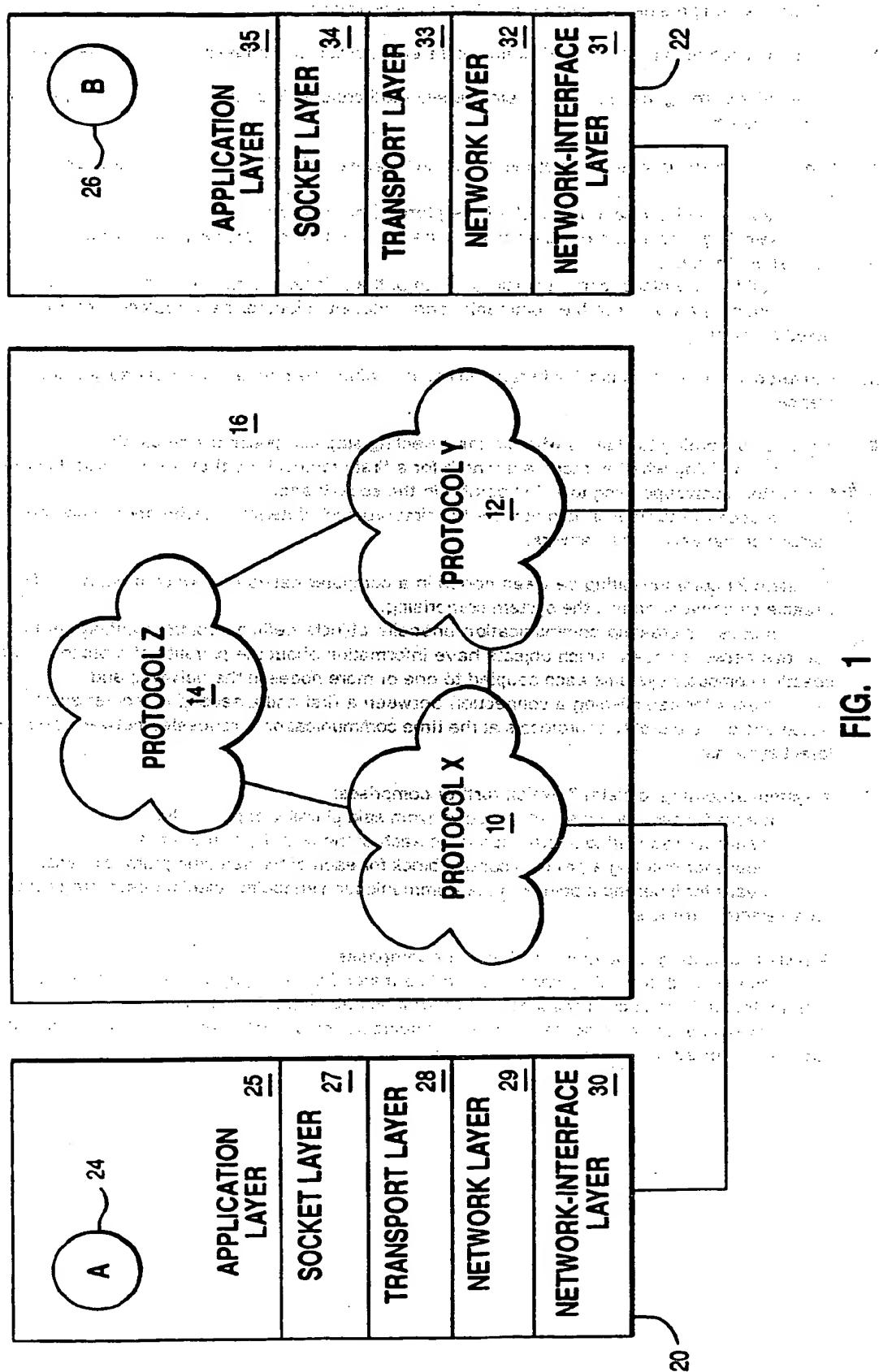
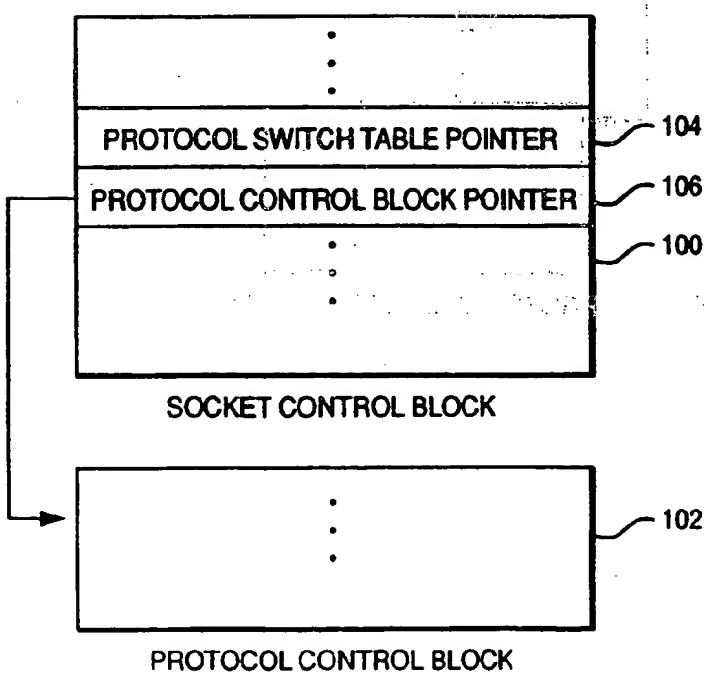
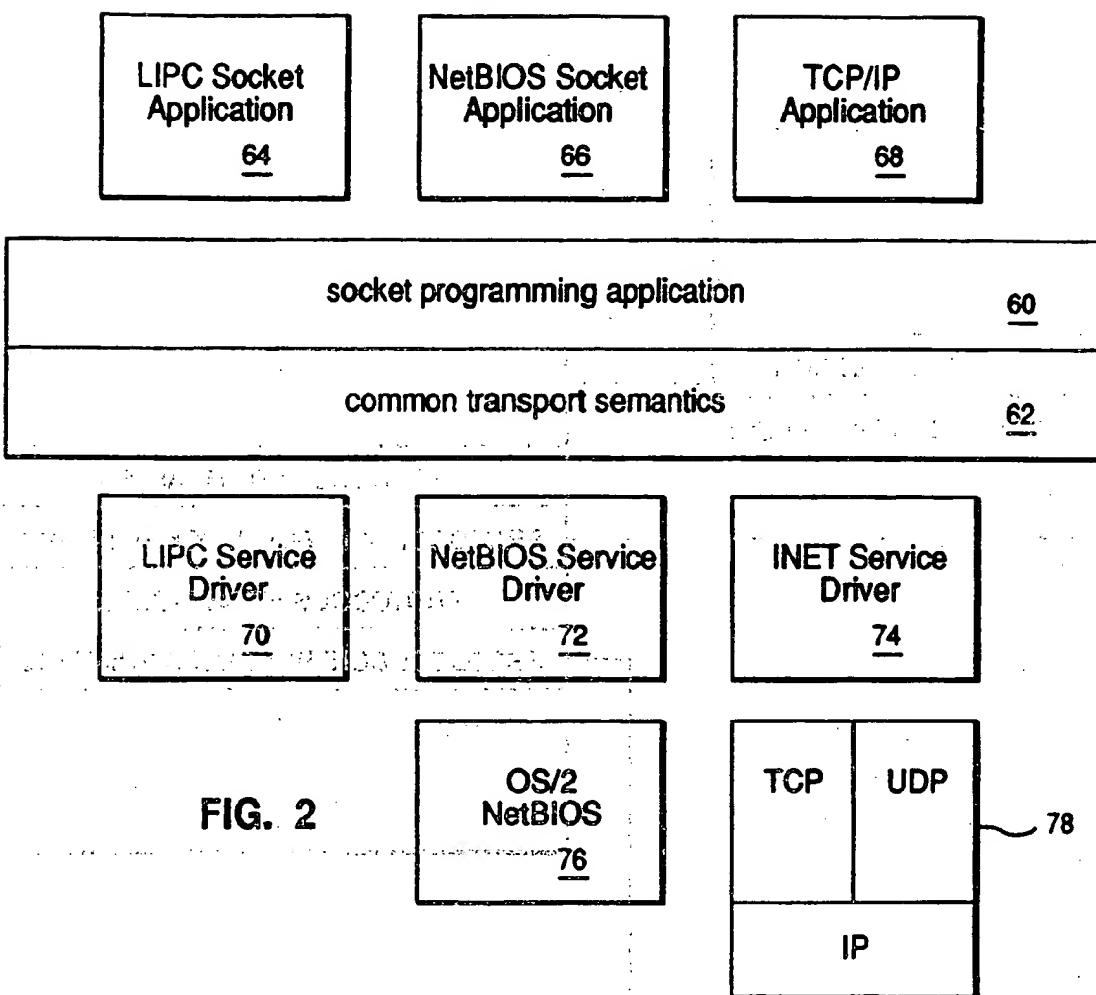


FIG. 1



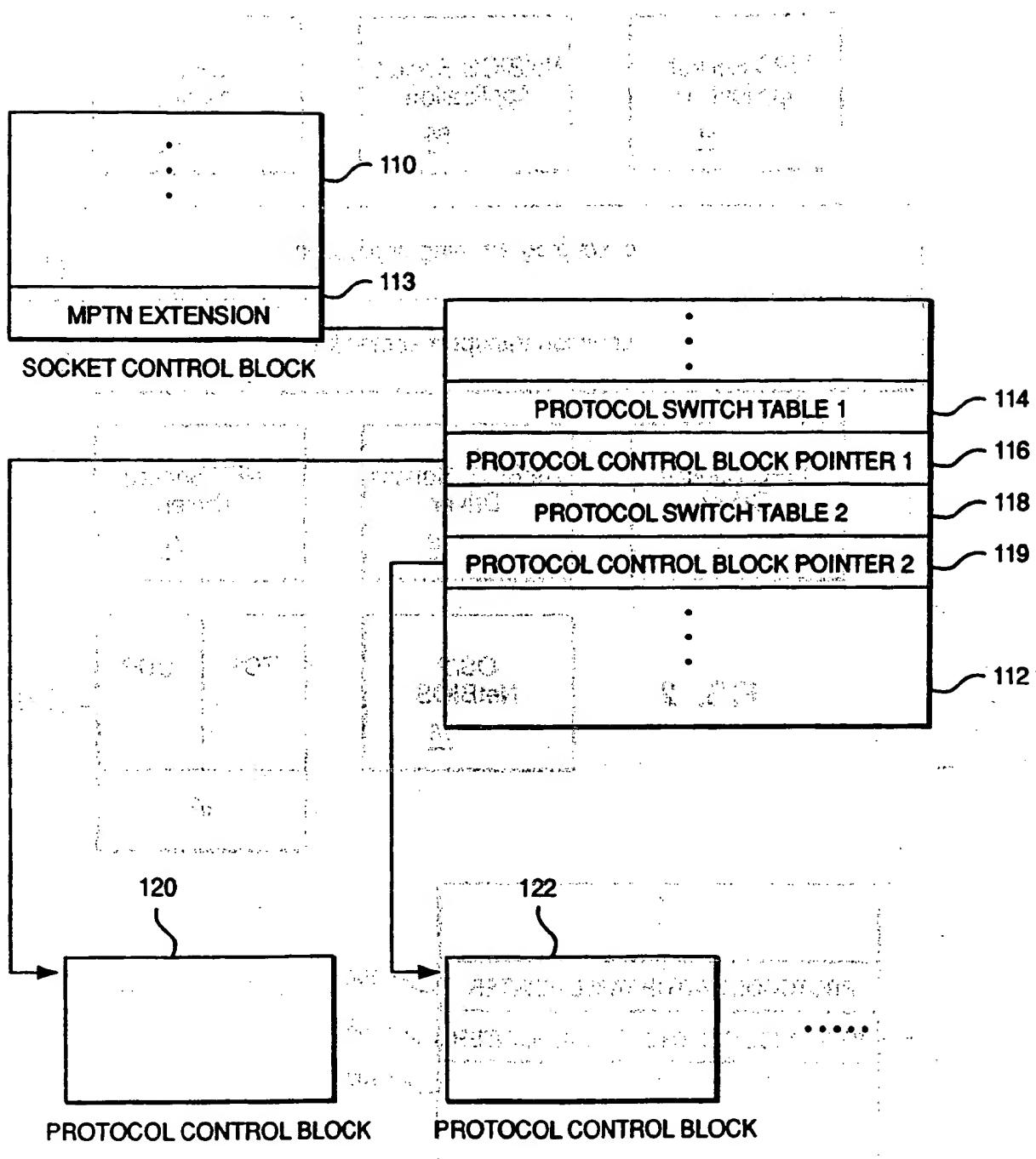
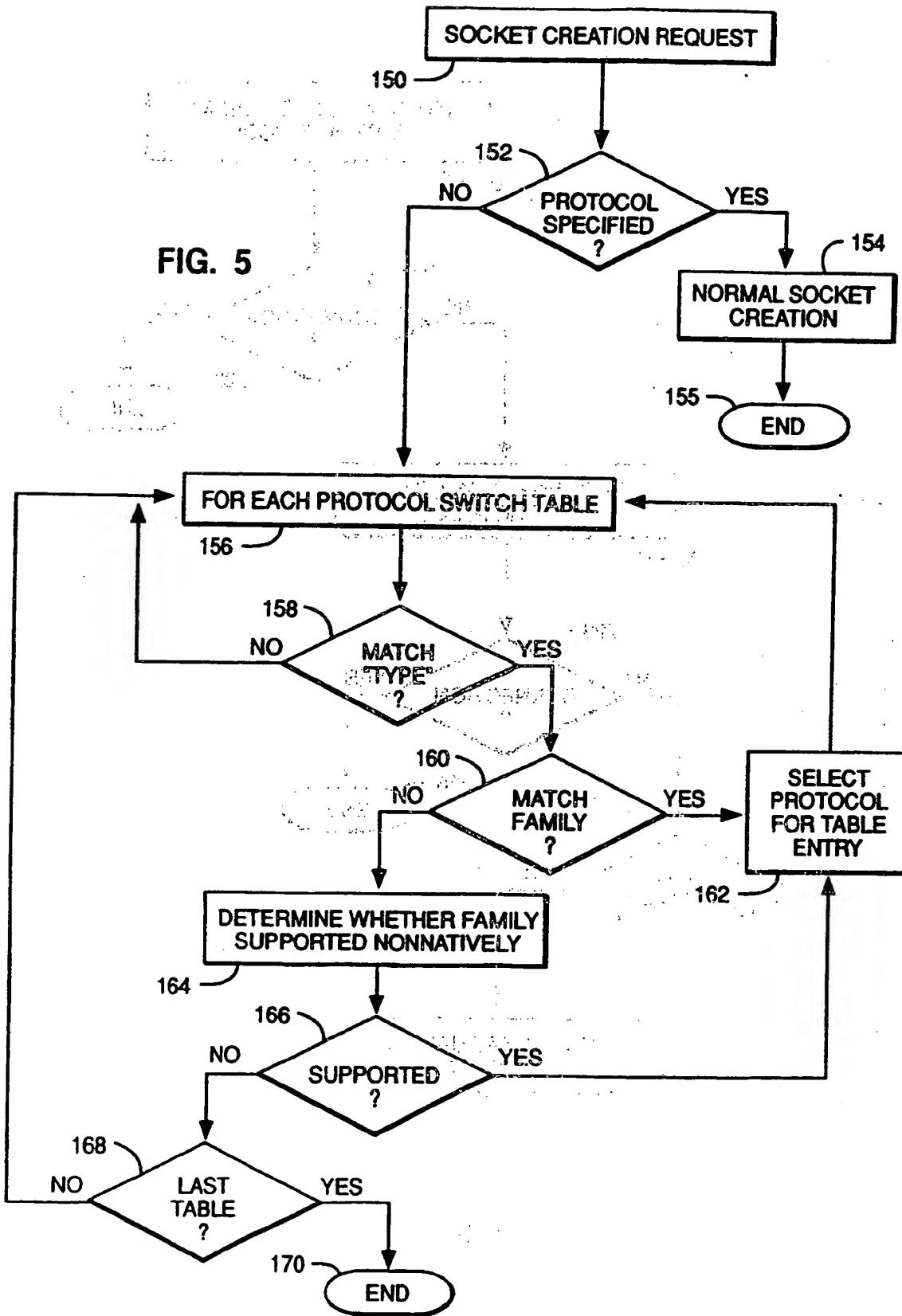


FIG. 4

FIG. 5



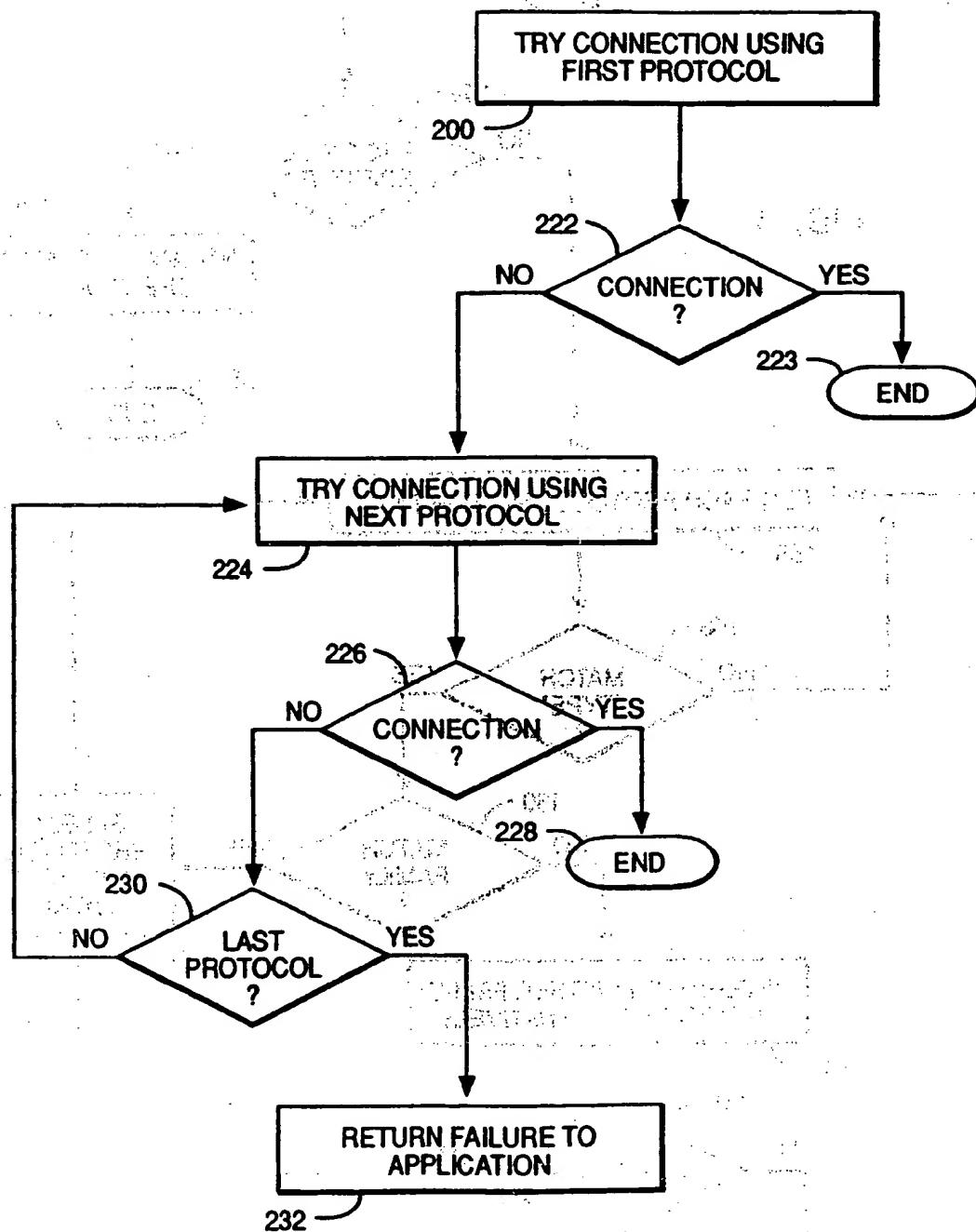
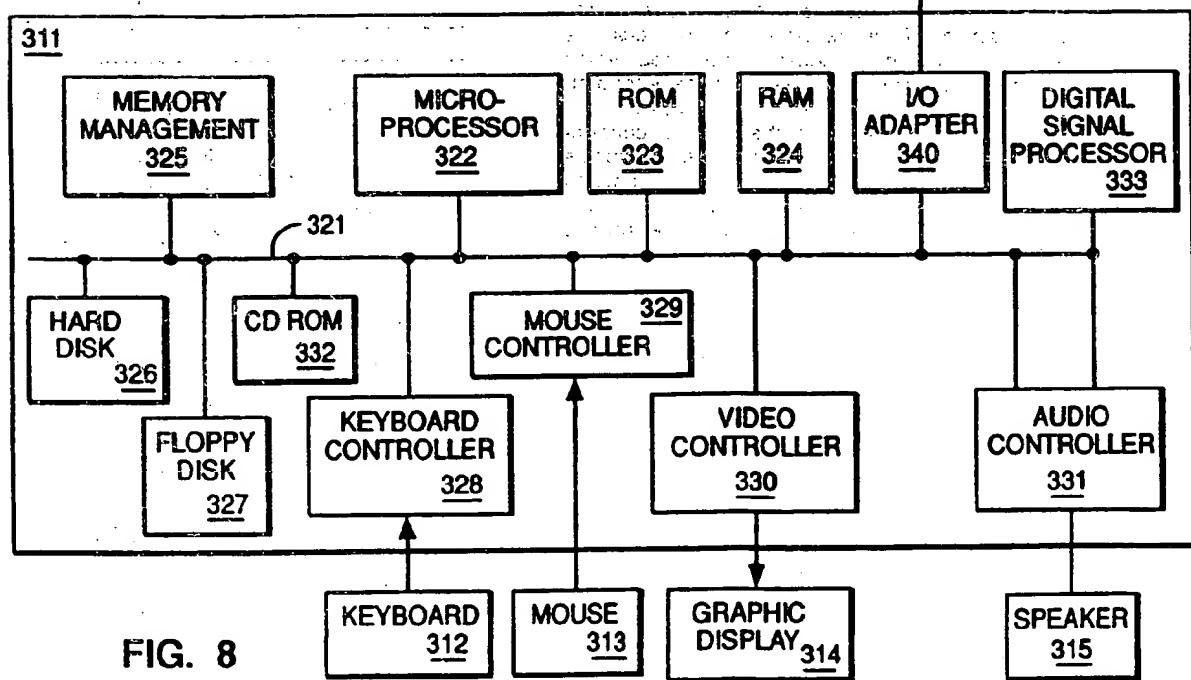
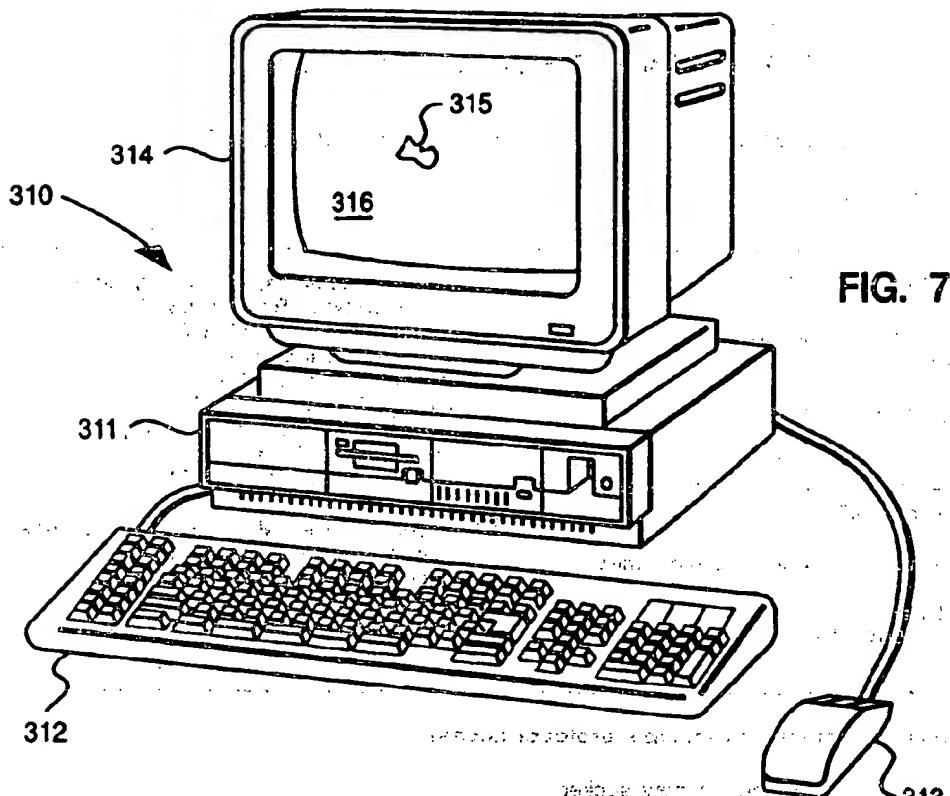


FIG. 6





European Patent
Office

EUROPEAN SEARCH REPORT

Application Number
EP 94 30 0677

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.)
Y	COMPUTER COMMUNICATIONS., vol.10, no.1, February 1987, GUILDFORD GB pages 21 - 29 D.COIFFIELD ET AL 'TUTORIAL GUIDE TO UNIX SOCKETS FOR NETWORK COMMUNICATIONS' * page 22, left column, line 35 - page 25, left column, line 7 *	1, 7	H04L29/06
Y	IEEE TRANSACTIONS ON SOFTWARE ENGINEERING., vol.17, no.1, January 1991, NEW YORK US pages 64 - 76 N.C.HUTCHINSON ET AL 'THE X-KERNEL: AN ARCHITECTURE FOR IMPLEMENTING NETWORK PROTOCOLS' * paragraph II *	1, 7	
A	IBM SYSTEMS JOURNAL., vol.27, no.2, 1988, ARMONK, NEW YORK US pages 90 - 104 M.S.KOGAN ET AL 'THE DESIGN OF OPERATING SYSTEM/2' * page 99, left column, line 17 - page 100, right column, line 35 *	1-9	TECHNICAL FIELDS SEARCHED (Int.Cl.) H04L

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	15 November 1994	Canosa Areste, C
CATEGORY OF CITED DOCUMENTS		
X : particularly relevant if taken alone Y : particularly relevant if combined with another document of the same category A : technological background C : non-written disclosure P : intermediate document		
T : theory or principle underlying the invention E : earlier patent document, but published on, or after the filing date D : document cited in the application L : document cited for other reasons & : member of the same patent family, corresponding document		

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- BLACK BORDERS**
- IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- FADED TEXT OR DRAWING**
- BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- SKEWED/SLANTED IMAGES**
- COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- GRAY SCALE DOCUMENTS**
- LINES OR MARKS ON ORIGINAL DOCUMENT**
- REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- OTHER:** _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

This Page Blank (USPTO)

This Page Blank (USPTO)